Using fuzzy modeling for consistent definitions of product qualities in requirements

Jean-Marc Davril, Maxime Cordy, Patrick Heymans, Mathieu Acher

Using Fuzzy Modeling for Consistent Definitions of Product Qualities in Requirements

Jean-Marc Davril
University of Namur, Belgium
jean-marc.davril@unamur.be

Maxime Cordy
and Patrick Heymans
University of Namur, Belgium
maxime.cordy|patrick.heymans@unamur.be

Mathieu Acher
Inria and Irisa, Université Rennes 1, France
mathieu.acher@inria.fr

***Abstract*—Companies increasingly rely on product differentiation and personalization strategies to provide their customers with an expansive catalog, and tools to assist them in finding the product meeting their needs. These tools include product search facilities, recommender systems, and product configurators. They typically represent a product as a set of features, which refer to a large number of technical specifications (e.g. size, weight, battery life). However, customers usually communicate and reason about products in terms of their qualities (e.g. ease-of-use, portability, ergonomics). In this paper, we tackle the problem of formalizing product qualities in the requirements of product-centred applications. Our goal is to extract product qualities from their technical features, so that customers can better perceive and evaluate the proposed products. To this end, we design a procedure for identifying segments of textual product documentation related to specific product qualities, and propose an approach based on fuzzy modeling to represent product qualities on top of technical specifications. Preliminary experiments we carried out on a catalog of cameras tend to show that fuzzy modeling is an appropriate formalism for representing product qualities. We also illustrate how modeled qualities can support the design of product configurators that are centered on the customers' needs.**

I. Introduction

In an ever more competitive environment, many companies identify opportunities for product differentiation and product personalization as a mean to retain and gain market share. These strategies can result in very large product assortments, which can suit the needs of an even larger number of customers. Yet there is a substantial risk of customers being overwhelmed by the number of alternatives to consider. Indeed, Jacoby et al. [1] show that information overload can prevent consumers from making efficient decisions. To reduce this risk, suppliers typically provide applications that help customers navigate through their offer with the aim of effectively and rapidly identifying the products that best fit their needs. These applications include catalog browsing interfaces, search engines, and recommender systems.

Similar concerns are encountered in product configuration systems. In [2] Franke and Schreier show that the enjoyment and perceived effort of the co-design process have a direct impact on the willingness to pay for customized products. In [3] Piller and Blazek state that a tedious co-design process can make customers reject a customization system.

Effective applications to support customers in navigating product catalog thus appear as an essential pre-requisite for successful and sustainable customization strategies. We name them *product-based application*. Therein, products are typically described as sets of *features*, i.e. functional or technical domain-specific concepts, and these descriptions are internally encoded in some formal representation such as *product matrices* [4]–[8]. In a product matrix, a row corresponds to a given product and a column to a given feature. A cell then gives the value of the feature in the specific product.

A limitation of feature-based approaches is that customers cannot always assess the appropriateness of products on the sole basis of their features [9]. They instead base their evaluation on the product *qualities*, i.e. criteria that together determine to which degree a given product can bring them satisfaction in regards to a particular use case. For example, a desired quality of a laptop is its portability. While a product matrix would represent features such as the dimensions, weight and battery life of the laptop, it would not give a direct representation of the quality of portability that can serve as a comparative measurement for the customers. There is thus a mismatch between how products are represented in the system, and the actual qualities that customers have in mind when evaluating the suitability of products.

While product matrices do not present direct measurements for product qualities such as *portability*, *performance* or *ease of use*, it is sometimes important to specify the requirements of product search facilities in terms of these qualities. Product qualities are the concepts which customers refer to for framing the suitability of the products to their needs. Therefore applications sometimes need to articulate user interactions around product qualities in order to meet the cognitive expectations of their users.

We aim at reducing the cognitive efforts of customers when searching a product based on expected qualities. As a first step, we present in this paper a method for augmenting product matrices with product qualities. Our approach, illustrated in Figure 1, relies on information retrieval from textual documentation and fuzzy modeling. More precisely, the two inputs of the method are textual product documentations and a product matrix. First, a supervised technique traces product qualities from textual product documentation. The results of this procedure then support the manual identification of dependencies between features and product qualities. Second, a linear regression model between the qualities and their dependent

features is computed. The regression model is then used to assess the qualities of products. Third, a set of *linguistic variables* is defined with the use of *fuzzy subsets* [10], [11]. Combining the assessment of product qualities with linguistic variables provides a formal model for reasoning about product qualities. It can also enable a product-based application to communicate with its users within a language that fits their cognitive expectations.

As shown in Figure 1, the purpose of the proposed method is to synthesize insights from both product specifications and domain knowledge into product-based applications.

Throughout the paper, we apply our procedure on a product matrix of 51 interchangeable lens cameras and a set of textual buying guides extracted from websites for camera enthusiasts. This preliminary application shows that dependencies between qualities and features retrieved from textual documentation can be used to infer a model which partially explains the variation of quality ratings in online available product reviews. Nonetheless, further evaluation with larger datasets will be required in future work to achieve higher statistical significance. Finally, we also discuss how including the representation of linguistic variables in *feature models* [12] can enable shifting from feature-centred to user-centred product configurators.

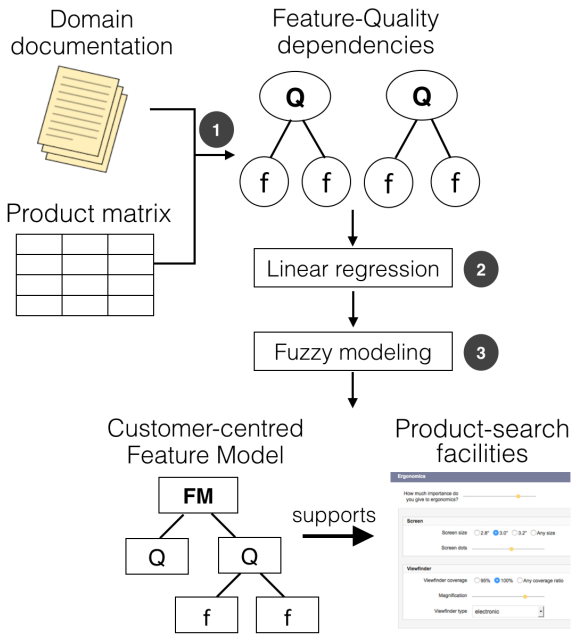

Fig. 1. Approach overview

The remainder of the paper is structured as follows. Section II introduces the necessary background. Section III describes our supervised method for tracing product qualities in textual product documentation. Section IV presents the modeling of linguistic variables on top of product matrices to represent product qualities. Section V discusses the use of linguistic variables to devise user-centered configurators. Section VI discusses threats to validity. Section VII presents related work while section VIII discusses future work.

II. Background

A. Product Matrix

A set of products can be represented within a *product matrix* [7]. Numerous organisations and companies create, maintain and provide matrices (e.g. for customers). A product matrix is a product-by-feature representation that describes a set of related products as sets of features. The content of the cell (i,j) represents the value of the j^{th} feature in the i^{th} product. Table II-A is a simplified example of a product matrix for a product line of laptops. Each column represents a feature and each feature has a type. Usual feature types are boolean, integer, real, string, and corresponding enumeration types. A product is thus defined as a set of values that are assigned to the features represented within the matrix.

	Price	**Screen size**	**HDMI output**	**Data storage**
P1	460	11.6”	yes	HDD
P2	485	10.1”	no	HDD
P3	899.99	12.6	yes	SSD
P4	1149.99	13”	yes	SSD

TABLE I
A PRODUCT MATRIX FOR A PRODUCT LINE OF LAPTOPS.

B. Feature Model

Feature Models (FM) have been first introduced for explicitly representing the differences and commonalities among the products of a product line [12]. An FM defines the valid combinations of features of a product line, each combination (sometimes called *configuration*) corresponding to an individual product of the line. An FM has a tree hierarchy in which nodes represent features and parent-child relationships define how features can be combined in product configurations.

Figure 2 shows a simplified example of an FM for a product line of laptops. The feature `Storage` is the parent of a *XOR-group* composed of the two features `HDD` and `SSD`. The XOR-decomposition specifies that exactly one of the child feature must be present in every product configuration. Other usual decomposition types are *OR-groups* and *Mutex-groups*, which respectively specify that when the parent feature is selected, all features, or at most one feature, must be included. Empty and full circle at the end of parent-child edges respectively represent optional and mandatory features. An FM is said to be attributed if attributes are associated to its features. The FM in Figure 2 shows two feature attributes: `Storage.capacity` and `Screen.size`. Finally, besides the constraints implied by its hierarchy, an FM can be complemented with additional constraints. The FM in Figure 2 is completed with the additional constraint: $\texttt{capacity} > 256 \Rightarrow \texttt{HDD}$.

The semantics of an FM fm, noted $[\![fm]\!]$, is commonly defined as the sets of products (i.e. configurations of features) that satisfy the constraints specified by fm [13]. Table II-B lists four valid product configurations for the FM in Figure 2. An FM can be used to communicate and formally reason about a set of products. It can also be the basis for devising *configuration interfaces* [14] (see e.g. Figure 7, page 7).

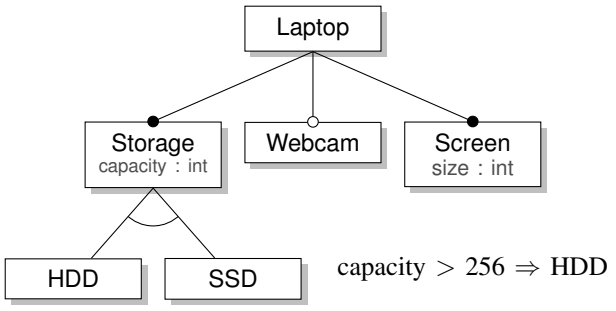

Fig. 2. A sample FM for a product line of laptops

	HDD	SSD	Storage capacity	Webcam	Screen size
P1	no	yes	128	yes	12"
P2	no	yes	256	yes	12.6"
P3	yes	no	2000	no	12"
P4	yes	no	2000	yes	14"

TABLE II
A SAMPLE SET OF VALID PRODUCTS FOR THE FM IN FIGURE 2.

C. Fuzzy Modeling

Zadeh [10] introduced the concept of a *fuzzy subset* as a generalization of ordinary crisp sets. Fuzzy subsets associate their elements to degrees of membership and are commonly used to represent propositions which truth value is not sharp. While ordinary sets can be seen as predicates whose truth values belongs to the set {0,1}, a fuzzy subset can be seen as a predicate which truth values are drawn from the interval [0,1] by a *membership function*.

Definition II.1 (Fuzzy subset)**.** *Let the set U be the universe of discourse. If X is a fuzzy subset of U, then X is associated with a membership function $\mu_X : U \rightarrow [0,1]$*

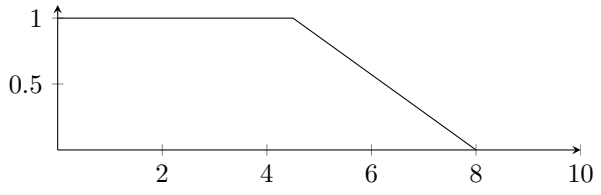

Fig. 3. The membership function for the fuzzy subset *lightweight*

Figure 3 shows the membership function defined in equation 1 for the fuzzy subset *lightweight* over the universe of discourse [0,10], which represents the weight of a laptop in pounds.

$$\mu_{lightweight}(u) \begin{cases} 1 & \text{if} \quad u <= 4.5 \\ 1 - \frac{u-4.5}{3.5} & \text{if} \quad 4.5 < u < 8 \\ 0 & \text{if} \quad u >= 8 \end{cases} \qquad (1)$$

As illustrated with the membership function in Figure 3, fuzzy subsets can be used to convey the meaning of concepts (i.e. linguistic values) such as *lightweight*. The function tells us that a laptop which weight is 3 lbs is definitely light, that a laptop of 4.5 lbs is lighter than a laptop of 6 lbs and that a laptop of 5.5 lbs is somewhat light.

A *linguistic variable* is a variable which values are linguistic values that denote fuzzy subsets [11]. For example if the linguistic variable L represents the concept of a laptop weight, then the statement *L is lightweight* indicates that the linguistic value *lightweight* is assigned to the linguistic variable *L*, and offers an imprecise indication about the actual value of the laptop weight. The *term-set* of a linguistic variable is the set of all its linguistic values. A possible term-set for L is {*lightweight*, *middleweight*, *heavyweight*}.

Definition II.2 (Linguistic variable)**.** *A linguistic variable V is a triplet (T, U, M) in which T is the term-set of V, that is, the set of all the possible linguistic values for V; U is the universe of discourse; and M is a semantic rule that associates each element of T to a fuzzy subset.*

Linguistic variables and fuzzy subsets allow the representation of uncertain knowledge about the exact values of variables, and fuzzy logic provides a formal framework to perform approximate reasoning on these values. A more comprehensive overview of fuzzy modeling can be found in [15].

III. TRACING PRODUCT QUALITIES IN TEXTUAL DOCUMENTATION

We are interested in creating traceability links between textual product documentation and product qualities. To this end, we extracted buying guides for cameras from *bestbuy.com*, *CNET.com*, *tomsguide.com*, *photographyconcentrate.com*, and *dpreviews.com*. Overall our documentation texts consists of seven buying guides (three from *dpreviews*, one per other website). Buying guides can be seen as being complementary to product matrices as they expose information that helps consumers relate technical features to product qualities. The buying guides contain a total of 25,652 words.

In this section, we present our results w.r.t. attempts to identify text segments that are related to the camera qualities `image quality`, `low light` and `ergonomics`. The qualities respectively represent the quality of image offered by the camera, its performance in low light settings and the overall ergonomics of its screen and viewfinder. These qualities were arbitrarily selected due to regular mentions in buying guides and product reviews.

The documents were preprocessed by filtering-out stopwords (i.e. very common words that do not cary meaningful information such as *the*, *a*, *which* or *at*). The remaining words were replaced by their lemmatized forms. For example, the words *send*, *sends*, *sent* and *sending* can all be replaced by their lemma *send*. We used the lemmatizer from the stanford-corenlp java library provided by the Stanford Natural Language Processing Group [1].

[1]See http://nlp.stanford.edu/software/corenlp.shtml

A. Product Quality Retrieval

In order to identify the relevant text segments, we relied on term-weighting within a contrastive method. First, the method needs to be provided with a set of *seed terms* about the product qualities that the user of the method expects to appear in relevant text segments. The method then presents the user with sentences that contain lots of these seed-terms, and the user manually classifies the highlighted sentences into true positives and false positives. After this manual evaluation, the method tunes the weights of the terms according to their number of occurrences in the true-positive sentences. Once the term weights have been adjusted, the method starts a new iteration and highlights sentences on the basis of the new weights. The method thus requires some level of domain knowledge from its user, as she is expected to provide the seed terms, and classify highlighted segments into true positives and false positives.

For each product quality, the method maintains a *vector space model* that associates terms with weights. The weights indicate the degree to which each term is representative of a that product quality. As first suggested by Salton et al. [16], we set the weights by computing a *tf-idf* (term frequency - inverse document frequency) metric:

$$w_{t,q} = tf_{t,q} \times \log \frac{N}{df_t} \tag{2}$$

where, $tf_{t,q}$ is the number of times that the term t occurs in the true-positive sentences for the quality q, df_t is the number of sentences that contain the term t, and N is the total number of sentences.

At each iteration, for each product quality q, the sentences that have not been highlighted yet are evaluated by the method. The evaluation of a sentence results in a score that indicates how likely it is related to q. We propose to compute the score for the sentence s wrt. to the product quality q as follows:

$$S_{s,q} = \sum_t w_{t,q} - w_{t,\neg q} \times (1 - \frac{|TP_q|}{|TP_q| \times |FP_q|}) \tag{3}$$

where t is a term in s, $w_{t,q}$ and $w_{t,\neg q}$ are the tf-idf scores (see Equation 2) for t w.r.t. the sentences that have been respectively classified as true positives and false positives during the previous iterations. TP_q and FP_q are respectively the set of all true positives and the set of all false positives for q that were found through the previous iterations. $\frac{|TP_q|}{|TP_q| \times |FP_q|}$ is thus the precision of the method in the identification of relevant sentences. The lower the current precision, the more the score is negatively affected when the sentence contains terms occurring in false-positive sentences. It allows the method to diverse its selection of sentences to present to users when the precision decreases.

B. Evaluation Results

Table III shows the evolution of precision and recall of our method on the set of buying guides for the product qualities `image quality`, `low light` and `ergonomics` through 25 iterations. Each time the user classifies an highlighted sentence as a true positive both precision and recall increase. Each time she classifies it as a false positive, the precision decreases. In order to compute these two measures, we manually tagged the sentences in the guides with the appropriate qualities before running the method. The manually tagged documentation thus served as the correct answer set, and was used to simulate the manual classification done by the user. At each iteration, 4 sentences were highlighted.

		#Iterations				
		5	**10**	**15**	**20**	**25**
`Image quality`	Precision	0.9	0.83	0.8	0.73	0.63
	Recall	0.24	0.45	0.65	0.78	0.85
`Low light`	Precision	0.95	0.88	0.73	0.61	0.51
	Recall	0.36	0.66	0.83	0.92	0.96
`Ergonomics`	Precision	0.8	0.73	0.63	0.59	0.47
	Recall	0.34	0.73	0.81	1	1
`Average`	Precision	0.88	0.81	0.72	0.64	0.54
	Recall	0.31	0.57	0.76	0.9	0.94

TABLE III
THE PRECISION AND RECALL FOR THE TRACING OF THREE PRODUCT QUALITIES THROUGH 25 ITERATIONS

Discussion. While the results show a promising average recall of 94% over 25 iterations, this initial evaluation does not take into account the sensitivity of the method wrt. the expertise of the current user who provides the seed terms and filter the highlighted sentences. Additionally, we did not evaluate the actual effort required from users when evaluating sentences, which makes difficult to assess the precision rate.

We can observe that, overall, the precision decreases as the number of iterations increases. An explanation to this trend is the need for false positives to adapt the term weights, as shown in Equation 3. The term weights are tuned at each iteration to guide the navigation of the space of all sentences. False positives are thus required to occur in order to adapt the navigation when most of the remaining undetected sentences are not well covered by the current vector space model.

The proposed approach can be applied on a large number of product qualities. While we arbitrarily limited our initial experiment to three of them, we did not discuss the identification of important product qualities (i.e. which are the qualities users care about?). This step requires to understand how users evaluate products. It would be interesting to complement our approach with the recommendation of product qualities. Similarly, while we let the user select the seed terms, recommending seed terms could improve the performance of our approach. Finally, in order to reduce manual effort, it would be interesting to study whether it is possible to consistently compute the optimal number of sentences to show the user, and to understand when the user can be recommended to stop classifying sentences.

IV. Definition of Linguistic Variables over Product Matrices

In this section we use the sentences identified in the buying guides as described in Section III to discover which features are related to particular product qualities. We represent product

qualities as linear combinations of these features. We then use linguistic variables to model the satisfaction of product qualities by individual products.

A. Dataset preprocessing

We applied our procedure to a dataset made of the technical specifications of 51 cameras that we extracted from the website *dpreview* [2]. The product matrix consists of 93 features for a total of 4743 cells. 619 cells (13.05%) are empty (i.e. the values for the corresponding features were missing in the product specifications). We then prune the matrix to the features relevant wrt. the product qualities (i.e. the features which values impact product qualities). We refer to these feature as the *explanatory features*. The task of identifying which features are explanatory must be supported by relevant domain knowledge. We used the sentences highlighted as described in Section III to manually identify which features are explanatory for a particular quality.

In order to apply a linear regression algorithm, all the cell values of the matrix are required to be numerical. Therefore, the nominal features are transformed into binary features mapped to values 0 and 1. This is achieved by replacing each k-valued nominal features by k binary features, one for each possible value v of the nominal feature, and indicating whether the feature has the value v or not. The result is a matrix of 27 features for a total of 1377 cells. 57 of the cells have missing values (3.41%). We thus impute a value for each empty cell by computing the average value for the corresponding feature over all the products.

B. Linear regression

We now compute a linear regression between the product qualities and the explanatory features. The linear regression is used to model the relationship between the qualities and the features. The resulting regression model is of the form $Q = F\beta + \epsilon$ as follows:

$$\begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{pmatrix} = \begin{pmatrix} f_{11} & \cdots & f_{1m} \\ f_{21} & \cdots & f_{2m} \\ \vdots & \ddots & \vdots \\ f_{n1} & \cdots & f_{nm} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

where q_i is the score of the i^{th} product for the quality q, and $f_{i,j}$ is the value of the j^{th} explanatory feature for the i^{th} product. In our experiment, the feature values ($f_{i,j}$) come from the product matrix, while the scores for the quality (q_i) comes from the ratings of q in product reviews on the *dpreview* website. The elements of the vector β are the *regression parameters*. This means that, given the values of the features and the scores of the quality in product reviews, computing the regression model consists in finding the vector β that best explains the relationship between the feature values and the quality scores. Finally, the elements of ϵ are the *error-terms* which describe the random component of the linear relationship between Q and F.

[2]http://www.dpreview.com

We use the Ordinary Least Squares (OLS) linear regression method for estimating the regression parameters in β. The OLS method attempts to minimize the Sum of Squared Residuals (SSR) which represents the difference between q_i, the actual rating of the quality, and $\hat{q}_i$, the value predicted by the model:

$$SSR = \sum_{i=1}^{n} (q_i - \hat{q}_i)^2 \tag{4}$$

By minimizing the SSR, the regression model captures the vector β that is best capable of predicting the score for the quality q of a given product p based on the feature values of p. The parameters in β can be seen as a model that has been learned from scores in product reviews, and that can be used to compute $q(p_i)$, which estimates the score of the product p_i for the quality q:

$$q(p_i) = \beta_1 f_{i1} + \cdots + \beta_m f_{im} + \varepsilon_i \tag{5}$$

Table IV-B shows the R-squared measure of the goodness of fit for the parameters in β. R-squared is defined as follows:

$$R^2 = 1 - \frac{SSR}{\sum_{i=1}^{n} (q_i - \bar{q}_i)^2} \tag{6}$$

where $\bar{q}_i$ represents the mean for q. The R-squared measure makes the SSR of the model relative to what it would have been if the average values of the features would have simply been used as predictors. When using the R-squared measure to evaluate the regression model, all the features of the model are assumed to be truly explanatory (i.e. they should explain the variation of q_i). Adding an extra explanatory feature to the model will always increase the value of the R-squared measure, even artificially. Therefore, table IV-B also shows the adjusted R-squared measure, which adjusts its value for the number of explanatory features in the model relative to the number of products in the product matrix:

$$\bar{R}^2 = R^2 - (1 - R^2)\frac{m}{n - m - 1} \tag{7}$$

where m is the number of explanatory features and n is the number of products in the product matrix.

Product quality	R-squared	Adjusted R-squared
`Image quality`	0.87	0.8
`Low light`	0.6	0.53
`Ergonomics`	0.76	0.71

TABLE IV
R-SQUARED AND ADJUSTED R-SQUARED MEASURES FOR THE REGRESSION MODEL

The R-squared measure must be interpreted as the degree to which the regression model explains the variability in the product ratings in comparison to a set of arbitrary predictors.

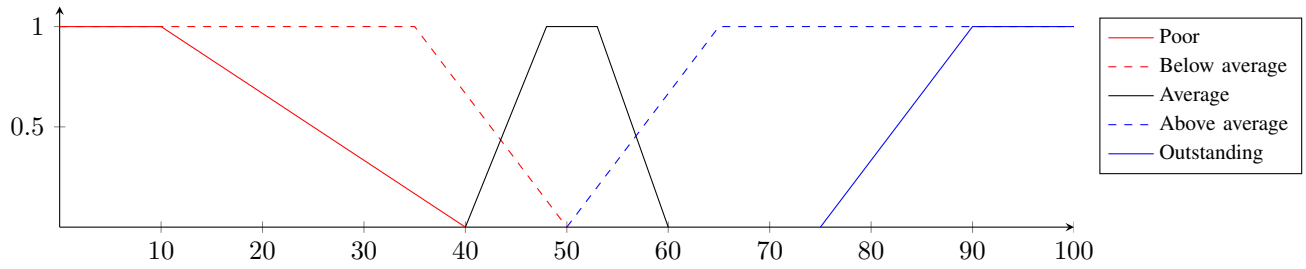

Fig. 4. Membership functions for the subsets of the linguistic values

C. Linguistic variables modeling

The regression model we previously learned can be used to predict the value of a quality for a given product. We use $q(p_i) \in [0, 100]$ to denote the value assessed by the regression model to q for the product p_i. For example, *image quality*$(p_i) = 85$ indicates that, based on the values of the features of p_i, the regression model estimates a score of 85 for the image quality of p_i.

We now define one linguistic variable for each product quality and 5 linguistic values common to all the qualities. The labels of the linguistic values are `poor`, `below average`, `average`, `above average` and `outstanding` and the membership functions of their associated fuzzy subset are shown in Figure 4. The statements in Figure 5 illustrate how the linguistic variables and their linguistic values can be used to describe a product p_i. For example, the statement over the linguistic variable `image quality` and the linguistic value `outstanding` indicates that the degree to which the image quality of p_i is outstanding is equal to 0.7.

$$\begin{array}{l} \text{image quality}(p_i) \text{ is above average} = 1 \\ \text{image quality}(p_i) \text{ is outstanding} = 0.7 \\ \text{low light}(p_i) \text{ is average} = 0 \\ \text{low light}(p_i) \text{ is above average} = 0.8 \end{array}$$

Fig. 5. Statements about the product qualities of p_i that rely on linguistic variables and their fuzzy subsets

The values of the statements are not probabilistic, but rather possibilistic, as first suggested by Zadeh [17], and indicative of a degree of truth. Formally, the degree of truth for the statement *image quality*(p_i) *is outstanding* is equal to $\mu_{outstanding}$(*image quality*(p_i)) where $\mu_{outstanding}$ is the membership function of the fuzzy subset associated to the linguistic value `outstanding` (see Figure 4).

V. Devising configuration interfaces from linguistic variables

To illustrate the use of linguistic variables in product-based applications, we discuss how product features and qualities can be used to write a product configuration model that is centered on customer needs. As suggested by Randall et al. [18], configuration systems should offer personalized user interactions based on the current user's level of expertise. The authors suggest the design of a *needs-based interface* for non-expert users and a *parameter-based interface* for performing configuration tasks on the technical features of the products.

Feature models can serve as configuration models and be used to derive configuration interfaces [14], [19]. Figure 6 shows an FM for the catalog of cameras. The first level of features under the root is made of the product qualities. The lower levels are comprised of the explanatory features for the qualities. As displayed in Figure 6, the FM focuses only on the branch that includes the feature `ergonomics`.

Figure 7 shows a possible user interface derived from the FM in Figure 6. The first element of the configuration interface allows the user to specify how much she values the product quality `ergonomics` (needs-based interface). The rest of the elements allow the user to configure the product features (parameter-based interface). If the user decides to perform the configuration task through the needs-based interface, the system can propose matching products based on the values of the linguistic variable for the quality `ergonomics`.

The system can also rely on the linguistic variables to inform the user on the availability of candidate products. For example, if the user indicates that she is interested in a camera with *outstanding image quality* and *outstanding portability*, the system might warn her that very few cameras satisfy both criteria and that she may need to lower her expectations in order to widen her search.

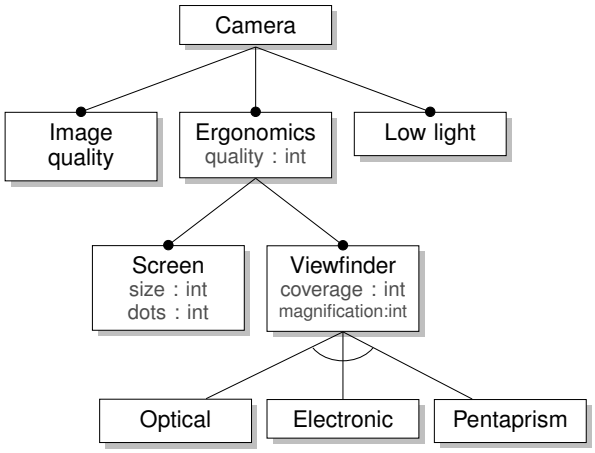

Fig. 6. Configuration model for the catalog of cameras

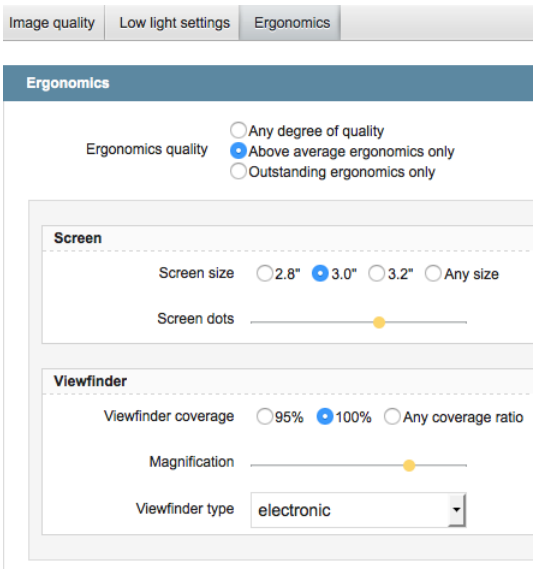

Fig. 7. Configuration interface derived from the FM in Figure 6

The use of linguistic variables to model product qualities suggests interesting possibilities of product-based applications that focus on user needs. We intend to investigate systematic methods to devise product search facilities from linguistic variables in future research.

VI. Threats to Validity

An *external threat* to validity is that we have only applied our procedure to one domain and one product matrix. It is still unclear whether our approach can be useful for formalizing product qualities in other domains and catalogs of products. We also retrieved information on product qualities from only one type of textual documents, namely buying guides for cameras. By definition, buying guides expose lots of information to help customers relate product characteristics to their needs. An interesting research direction would be to determine how the nature of the available textual documentation affects the applicability and usefulness of our approach.

Moreover, we did not address the problem of identifying which are the qualities that are important to customers. The user is responsible for listing the qualities, and our approach is thus sensitive to her degree of domain knowledge.

There are several *internal threats* to validity that are related to the use of OLS regression analysis. First, a linear regression assumes the possibility to define product qualities as linear combination of features. Linear regression gives a very first insight for explaining some product qualities. Yet perceptions of qualities by consumers are much more complex phenomena. Another threat is that the quality of the regression model can decrease when the number of products is limited and the number of explanatory features is high. Additionally, if wrong features are selected to explain the product qualities, the quality of the regression model also decreases. Finally, an important threat to validity is the limited size of the product matrix used in our experiment. We plan to assess the appropriateness of our approach with larger datasets in future work.

VII. Related work

Previous works have addressed the extraction of FMs from both product specifications [4], [5], [8], [20]–[22] and textual documentation [6], [23], [24]. Features of the synthesized FMs usually refer to functional aspects (i.e. technical features) of a product line. In our context we aim to infer product qualities and relate them to features. The intended benefit is that users can manipulate high-level concepts of an FM when configuring or choosing a product. Than et al. [25] propose an approach to relate an FM to requirements in order to enable stakeholders to derive optimal configurations at the requirements level. Their work focuses on well-defined, sharp requirements while we are interested in reasoning on high-level product qualities that are somewhat characterized by uncertain definitions.

Fuzzy logic has been used for modeling perceived product qualities and prioritizing product development requirements [26]–[28]. In [29] Tsai et al. define fuzzy inference rules to describe the relationship between customer needs and product features. The authors propose an optimum-searching method for identifying the combination of features that bests suits the particular needs of a customer based on the fuzzy inference rules. The focus of this work is to (1) mine technical features, product qualities, as well as their relationships from different artefacts (product specifications, comparison guides); (2) model the inferred information in an FM, and finally (3) devise product-search facilities for easing the task of customers.

In [30], a data mining technique (Range Ranking) is utilized to identify the most critical decisions in product configuration. In [4], probabilistic FMs are introduced and formally defined. Robak et al. [31] uses fuzzy logic in FMs to represent feature development priorities. Pieczyriski et al. [32] uses fuzzy logic to model customer behaviours and market events in an FM. In our case we use fuzzy logic to relate product qualities to technical features; we then encode the information in an FM.

Many researchers have relied on the Vector Space Model method to retrieve information from textual documents , e.g. see [33]–[35]. As part of our initial experiments we use similar techniques for tracing product qualities (see Section III).

VIII. Conclusion and future work

We presented a procedure for extracting product qualities in textual documents and for relating them to technical features We used linear regression and fuzzy modeling to represent the product qualities. We discussed the design of user-centered product configurators with a feature model containing linguistic variables representing qualities. Preliminary experiments we carried out on a catalog of cameras tend to show that fuzzy modeling can support the design of product configurators that are centered on product qualities. Hence users can manipulate and perceive products in terms of qualities such as *ergonomics* or *image quality* instead of rather technical features.

An important question is whether the identification of important product qualities and their relationships to features

can be covered across various domains by an extension of the approach proposed in Section III, or if domain specific methods should be designed depending on the availability and quality of product documentation. Furthermore we intend to formally evaluate the usefulness of fuzzy models of product qualities to guide users towards adequate products.

Fuzzy modeling for product qualities brings forward interesting research directions. A first research direction is the *reverse engineering* of linguistic variables from textual documentation with the aim of modeling product qualities. Today consumers can seek information about products from many different sources such as product reviews and ratings, QA lists, product comparison guides or discussion forums. This means that there are a lot of product information available online that can be analyzed to build valuable models about consumers and product qualities.

A second research direction is the *forward engineering* of product search facilities that rely on linguistic variables and approximate reasoning. More specifically, we plan to investigate the development of methods and tools to engineer user-centered configurators. A particularly interesting problem is the automatic search of satisfying product configurations from user interactions on the product qualities. This mechanism could allow non-expert users to specify their personal needs in terms of product qualities, and to be recommended with candidate matching products. The application of such navigation-based recommendations in configurators remains a major research challenge [36].

REFERENCES

[1] J. Jacoby, D. E. Speller, and C. A. Kohn, "Brand choice behavior as a function of information load," *Journal of Marketing Research*, 1974.

[2] N. Franke and M. Schreier, "Why customers value self-designed products: The importance of process effort and enjoyment*," *Journal of Product Innovation Management*, vol. 27, no. 7, pp. 1020–1031, 2010.

[3] F. Piller and P. Blazek, "Core capabilities of sustainable mass customization," *Knowledgebased Configuration–From Research to Business Cases. Morgan Kaufmann Publishers, Waltham, MA*, pp. 107–120, 2014.

[4] K. Czarnecki, S. She, and A. Wasowski, "Sample spaces and feature models: There and back again," in *Proceedings of the 12th International Software Product Line Conference, 2008. SPLC'08.*

[5] E. N. Haslinger, R. E. Lopez-Herrejon, and A. Egyed, "On extracting feature models from sets of valid feature combinations," in *Fundamental Approaches to Software Engineering*. Springer, 2013, pp. 53–67.

[6] J.-M. Davril, E. Delfosse, N. Hariri, M. Acher, J. Cleland-Huang, and P. Heymans, "Feature model extraction from large collections of informal product descriptions," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*.

[7] G. Bécan, N. Sannier, M. Acher, O. Barais, A. Blouin, and B. Baudry, "Automating the formalization of product comparison matrices," in *29th IEEE/ACM International Conference on Automated Software Engineering (ASE'14)*, sep 2014.

[8] G. Bécan, R. Behjati, A. Gotlieb, and M. Acher, "Synthesis of attributed feature models from product descriptions," in *19th International Software Product Line Conference (SPLC'15)*, Nashville, TN, USA, jul 2015, (research track, long paper).

[9] V. A. Zeithaml, "Consumer perceptions of price, quality, and value: a means-end model and synthesis of evidence," *The Journal of marketing*, 1988.

[10] L. A. Zadeh, "Fuzzy sets," *Information and control*, vol. 8, no. 3, 1965.

[11] L. A. Zadeh, *The concept of a linguistic variable and its application to approximate reasoning*. Springer, 1974.

[12] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (foda) feasibility study," DTIC Document, Tech. Rep., 1990.

[13] P. Schobbens, P. Heymans, and J.-C. Trigaux, "Feature diagrams: A survey and a formal semantics," in *14th IEEE international conference on Requirements Engineering*. IEEE, 2006, pp. 139–148.

[14] Q. Boucher, G. Perrouin, and P. Heymans, "Deriving configuration interfaces from feature models: A vision paper," in *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*. ACM, 2012, pp. 37–44.

[15] R. R. Yager and D. P. Filev, *Essentials of Fuzzy Modeling and Control*. New York, NY, USA: Wiley-Interscience, 1994.

[16] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, Nov. 1975. [Online]. Available: http://doi.acm.org/10.1145/361219.361220

[17] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets Syst.*, vol. 100, pp. 9–34, Apr. 1999.

[18] T. Randall, C. Terwiesch, and K. T. Ulrich, "Principles for user design of customized products," *California Management Review*, 2005.

[19] M. Schlee and J. Vanderdonckt, "Generative programming of graphical user interfaces," in *Proceedings of the working conference on Advanced visual interfaces*. ACM, 2004, pp. 403–406.

[20] S. She, R. Lotufo, T. Berger, A. Wasowski, and K. Czarnecki, "Reverse engineering feature models," in *Proceedings of the 33rd International Conference on Software Engineering (ICSE)*. IEEE, 2011, pp. 461–470.

[21] G. Bécan, M. Acher, B. Baudry, and S. Ben Nasr, "Breathing ontological knowledge into feature model synthesis: An empirical study," *Empirical Software Engineering (ESE)*, 2015.

[22] S. She, U. Ryssel, N. Andersen, A. Wasowski, and K. Czarnecki, "Efficient synthesis of feature models," *Information and Software Technology*, vol. 56, no. 9, pp. 1122–1143, 2014.

[23] N. Weston, R. Chitchyan, and A. Rashid, "A framework for constructing semantically composable feature models from natural language requirements," in *Proceedings of the 13th International Software Product Line Conference*. Carnegie Mellon University, 2009, pp. 211–220.

[24] A. Ferrari, G. O. Spagnolo, and F. Dell'Orletta, "Mining commonalities and variabilities from natural language documents," in *Proceedings of the 17th International Software Product Line Conference*. ACM, 2013.

[25] T. Than Tun, Q. Boucher, A. Classen, A. Hubaux, and P. Heymans, "Relating requirements and feature configurations: A systematic approach," in *Proceedings of the 13th International Software Product Line Conference*. Carnegie Mellon University, 2009, pp. 201–210.

[26] C. Temponi, J. Yen, and W. A. Tiao, "House of quality: A fuzzy logic-based requirements analysis," *European Journal of Operational Research*, vol. 117, no. 2, pp. 340–354, 1999.

[27] C. Kwong and H. Bai, "A fuzzy ahp approach to the determination of importance weights of customer requirements in quality function deployment," *Journal of intelligent manufacturing*, 2002.

[28] L.-H. Chen and M.-C. Weng, "A fuzzy model for exploiting quality function deployment," *Mathematical and Computer Modelling*, 2003.

[29] H.-C. Tsai and S.-W. Hsiao, "Evaluation of alternatives for product customization using fuzzy logic," *Information Sciences*, 2004.

[30] A. S. Sayyad, H. Ammar, and T. Menzies, "Software feature model recommendations using data mining," in *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering*, ser. RSSE '12. Piscataway, NJ, USA: IEEE Press, 2012.

[31] S. Robak and A. Pieczynski, "Employing fuzzy logic in feature diagrams to model variability in software product-lines," in *Proceedings of the 10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, 2003.

[32] A. Pieczyriski, S. Robak, and A. Walaszek-Babiszewska, "Features with fuzzy probability," in *Proceedings of the 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, 2004.

[33] M. W. Berry, Z. Drmac, and E. R. Jessup, "Matrices, vector spaces, and information retrieval," *SIAM review*, vol. 41, no. 2, pp. 335–362, 1999.

[34] J. H. Hayes, A. Dekhtyar, and J. Osborne, "Improving requirements tracing via information retrieval," in *Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International*. IEEE, 2003, pp. 138–147.

[35] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram, "Advancing candidate link generation for requirements tracing: The study of methods," *Software Engineering, IEEE Transactions on*, vol. 32, no. 1, pp. 4–19, 2006.

[36] J. Tiihonen, A. Felfernig, and M. Mandl, "Personalized configuration," *Knowledge-based Configuration–From Research to Business Cases. Morgan Kaufmann Publishers, Waltham, MA*, pp. 167–179, 2014.